AD-A209 606

Multicast Communication in Multiprocessor Systems

by

Gregory T. Byrd and Nakul P. Saraiya

Department of Computer Science

Stanford University
Stanford, California 94305

89   6   29   184

Multicast Communication in Multiprocessor Systems

Gregory T. Byrd and Nakul P. Saraiya

Stanford University
Stanford, CA 94305

Bruce A. Delagi

Digital Equipment Corporation
Palo Alto, CA 94301

Multicast Communication in Multiprocessor Systems *

Gregory T. Byrd and Nakul P. Saraiya
Knowledge Systems Laboratory
Stanford University, Stanford, CA 94305

Bruce A. Delagi
Digital Equipment Corporation
Palo Alto, CA 94301

Abstract

Recent high-performance multiprocessors exploit cut-through routing, with packets routed as their first bytes arrive. Hardware-supported multicast can benefit the many parallel programs in which producers provide each value to multiple consumers. We describe several cut-through multicast protocols, including a restrictive (yet adaptive) routing scheme for deadlock avoidance. Simulations using synthetic and application-driven loads show it has significantly better performance than either multicast emulation or deadlock detection and resolution. The scheme provides cut-through multicast without requiring dedicated storage in the communication facilities for a full packet. We thus extend ideas considered for efficient cut-through routing in multiprocessor systems to include multicast.

Keywords: multicast, cut-through routing, multiprocessors, communications networks, deadlock avoidance

1

1 Approaches to Multicast Transmission

The communication patterns naturally found in parallel programs include those in which a producer provides values to more than one consumer [2,5]. These patterns can be directly supported by communication facility routing protocols or indirectly supported by arranging that either the operating system or the application build a tree of communicating processes whose leaves are the "real" targets of communication. Intermediate nodes of this tree store packets and forward them to the next level of the tree. Such store and forward techniques may take limited advantage of available network facilities in a system providing cut-through routing [8]. Additionally, interrupt handling (and possibly process switching) latencies are incurred at each forwarding step. These can be significantly larger than the transmission time of the packet itself. These performance considerations motivated us to study means to provide direct support for multicast communication.

1.1 Suppose We Just Had Big Buffers?

The unicast communication protocol underlying the multicast facilities discussed here has been described in [1]. It includes provision for adaptive routing and is deadlock-free so long as the *computing nodes* (see figure 1) of the system have sufficient available storage to hold the blocked packets. The communication facilities themselves provide only enough buffering at each *port* to hold a packet target address. Flow control is done in units of transmission activity equal to one of these buffers. Independent routing decisions based on local path availability information are made by each *router* encountered by a packet as its front edge makes progress from its source to its target.

For normal operation, only small amounts of dedicated resources are used in the transmission and reception of information, namely the small amount of buffering provided at each port. In the exceptional condition that no suitable output port is available for an incoming packet, additional storage is made available by con-
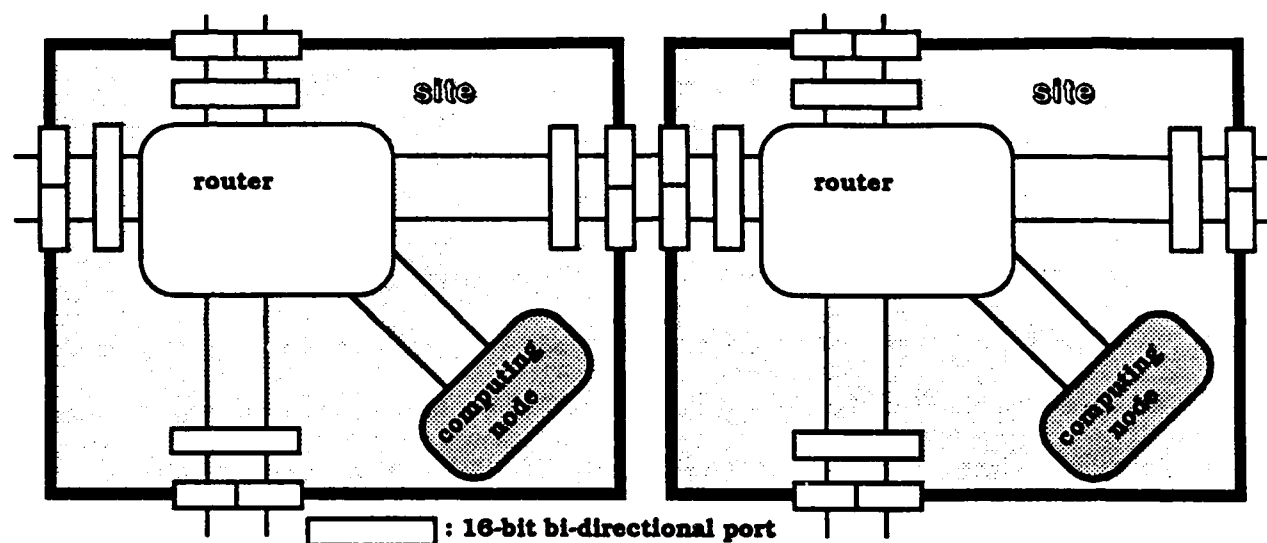
Figure 1: Sites—Computing Nodes, Ports, and Routers

tending for it from a buffer pool managed by the local computing node.[1] The packet is then stored for

forwarding to its target at a later time. The use of small dedicated buffers for normal operation together

with larger buffers allocated from a common pool of storage to handle exceptional conditions permits high

performance with simple, low cost, implementations.

If direct support of multiple consumer communication patterns in concurrent programs is to be provided,

the possibility of deadlock must be considered. As shown in figure 2, when two multicasts have each acquired

some paths also needed by the other, each will block progress by the other: a deadlock is the result.

One way to handle deadlock is to ensure that no path may be blocked; that is to associate with each input

port enough dedicated storage to buffer the largest possible message. Motivated as discussed above, however,

our design goal was for the multicast communication facility to have little dedicated storage relative to

---

[1] Through the use of virtual channels [4] we could eliminate the requirement for such buffers at the cost of consuming network resources holding delayed packets in the net and with significant impact on the possibilities for adaptive routing.
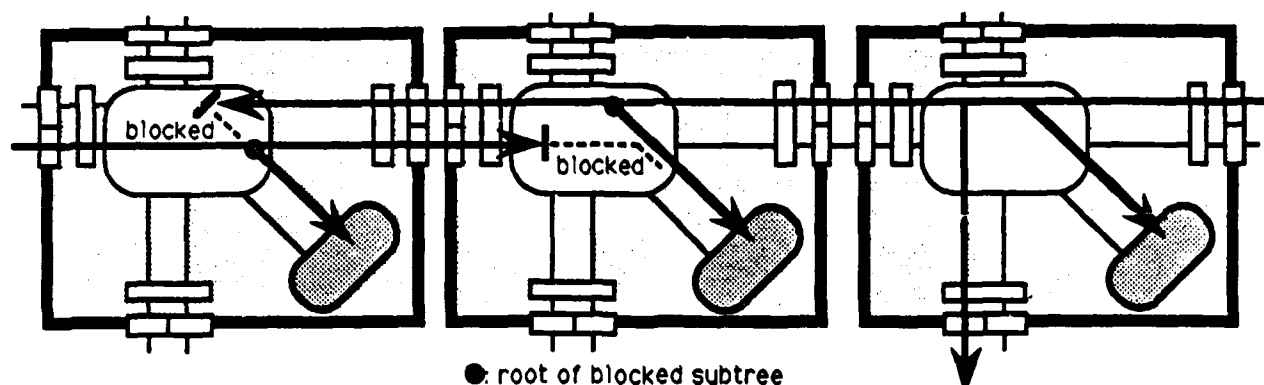
●: root of blocked subtree

Figure 2: Multicast Deadlock

anticipated packet sizes.[2] We were looking for a way to bring together deadlock-free multicast capabilities as proposed for store-and-forward networks [7] with concepts for efficient cut-through routing in multiprocessor systems [1,3]. We therefore chose to look beyond a brute force solution to the problem of providing efficient multicast.

1.2   But How Can Deadlock Be Handled?

We will consider several alternative approaches for dealing with the issue of deadlock in multicast transmission: (1) emulating multicast by multiple unicasts—eliminating the problem by effectively eliminating the facility, (2) detecting and resolving deadlocks, or (3) avoiding deadlock by restricting routing alternatives. We call the first approach *multi-unicast*, (MU). For the second approach, the deadlock detection and resolution scheme we consider allows all but the potentially deadlocked subtree to proceed. The transmission of just this subtree is aborted and later resumed. (Two such subtrees are shown in figure 2). We call this scheme *resumable multicast*, (RM), to distinguish it from the naive multicast implementation, *abortable mul-*

---

[2] As indicated, the dedicated storage in the transmission path need only be large enough to hold a target address. Breaking packets into sub-packets, introduces significant partition and re-assembly overhead and entails substantial proportionate increase in such dedicated storage.

ticast. The performance of abortable multicast—in which detection of the potential deadlock of any subtree of a multicast transmission results in aborting the entire multicast—was so poor that we will not consider it further in this paper. The final scheme discussed in this paper, providing deadlock avoidance by restricting routing alternatives, we term *restricted branch multicast,* (RBM).

1.3 What Will Be Shown?

In the remainder of this paper we will describe the implementation of each of the three schemes and present their simulated performance. Our simulation studies include both test cases contrived to exercise multicast facilities and also an application characterized by a mix of multicast and unicast communication. Based on our expectations of the relative times required to drive signals within and between sites in multiprocessor systems, the computing nodes in these simulations execute instructions four times faster than the network ports accept, transmit, and deliver input. Based on expected die pin count limitations, four pairs of 16-bit network ports are associated with each site. The computing node at each site is assumed to include a resource for message handling (including forwarding) that is independent of the resource doing computation.

We will discuss the performance of the communications system in each of the three schemes in terms of the measured *network latency* for the packets it handles. Network latency is defined in this paper as the difference between the time a packet is made available for transmission and the time it is completely received at its target computing node. If, due to unavailability of a suitable output port, a packet is temporarily stored at a computing node which is not its target, the time taken to forward the packet (as well as the time spent waiting to be forwarded) is included in the network latency for that packet. By this measure, the restricted branch approach to multicast transmission (RBM) will be shown in many cases to have significantly better performance than the other two schemes considered.

2   Multicast Protocols

In this section, we describe three approaches for supporting multicast. We give expressions for the average network latency for the three schemes in the absence of contention. Then, in order to assess the performance of each approach in the presence of network load, we compare the latencies experienced while running a test program designed to exercise the multicast facilities.

The latency of a *unicast* packet in a cut-through network, in the absence of contention, is

$$T = \left\lceil \frac{t}{W} \right\rceil D + \left\lceil \frac{L}{W} \right\rceil,$$

where:

t is the number of bits in a *target*, i.e., a network address;

D is the *distance* (the number of channels) traversed by the packet;

L is the *length* of the packet (data only), in bits; and

W is the *width* of the network channels, in bits.

The above equation gives latency in terms of *cycles*, where a cycle is the time to transfer W bits of data from an input port of one site to an input port on the next site. In describing the latencies experienced by the multicast schemes, we will use the symbols $\bar{D}$ and $\bar{L}$ to denote the *average* distance and length, respectively.

2.1   Multi-Unicast

The simplest form of support for multicast, the *multi-unicast* (MU) protocol, emulates it by transmitting the entire packet sequentially to each target.

In an MU transmission with n targets, the average packet must wait for the $(n-1)/2$ packets ahead of it to be delivered to the network. This time is $\frac{1}{2}(n-1)(\lceil t/W \rceil + \lceil \bar{L}/W \rceil)$, since both the data and a single target must be sent for each packet. Adding this to the network delay experienced by the average packet

yields

$$T_{\mathrm{MU}} = \left\lceil \frac{t}{W} \right\rceil \bar{D} + \left\lceil \frac{\bar{L}}{W} \right\rceil + \frac{n-1}{2} \left\lceil \frac{t}{W} \right\rceil + \frac{n-1}{2} \left\lceil \frac{\bar{L}}{W} \right\rceil .$$

Thus for large n or $\bar{L}$, the cost of placing multiple copies of the packet onto the network is likely to dominate the average latency.

2.2 Resumable Multicast

The MU approach creates n independent copies of a packet to be sent to n destinations. If the transmission paths between the source and several destinations have some set of channels in common, the MU approach uses those channels inefficiently by transmitting the same data multiple times. A more efficient approach might be to transmit along a common path as much as possible, then "split" the packet by transmitting the data down several paths simultaneously.

This is the approach taken by the *resumable multicast* (RM) scheme. Because of the limited buffering at network ports, data can only be transmitted if all paths are able to receive it. If one path is blocked, it must become unblocked before any of its siblings may proceed. As described earlier, this can lead to deadlock (see figure 2). Neither multicast can proceed, since each has acquired resources needed by the other. In RM, each site at which a split has occurred can detect a potential deadlock situation (e.g., if the number of consecutive blocked cycles has exceeded some threshold) and request that all the downstream transmissions from that point be aborted. A packet is aborted by immediately terminating it with a special character which tells the receiving site to "ignore this packet."

To recover from the abort, a local copy of the packet is kept and is retransmitted at a later time. (As with the aborted packet, a special packet terminator is used to indicate whether the temporarily buffered copy should be retransmitted or forgotten.) Since the network ports themselves do not have sufficient storage to temporarily buffer the packet, space must be allocated from the buffer pool of the site's computing node

whenever a packet is split. If space is not available, or if access to the computing node is blocked, the packet is not split but merely "passed through" toward the first target in the packet's target list.

In the absence of contention, the average RM latency is very close to the unicast latency. However, an RM packet contains $n - 1$ additional targets, relative to a unicast packet. Thus, the average latency equation is

$$T_{\text{RM}} = \left\lceil \frac{t}{W} \right\rceil \bar{D} + \left\lceil \frac{L}{W} \right\rceil + (n - 1) \left\lceil \frac{t}{W} \right\rceil.$$

The targets are now transmitted in series with the data, rather than sending a copy of the data with each target. The number of targets, then, is an *additive* component in the latency, rather than *multiplicative*, as we found in the MU case.

However, RM multicasts are very sensitive to network load, for three reasons. First, they split aggressively, thus increasing the chances that one branch will be blocked due to contention with some other packet in the network. Second, each split consumes the port connecting to the computing node at a site, regardless of whether there are any local targets, this precludes any other *useful* packet (multicast or unicast) from being accepted at that site. Third, *all branches* of a blocked subtree are aborted and retransmitted, thus wasting the resources they have seized as well as excluding other messages from using them. Aborts thus add to the network congestion, which can cause more aborts—positive feedback which can degrade the performance of the whole network.

2.3 Restricted Branch Multicast

The possibility of severe congestion problems with RM leads us to consider a third multicast protocol, the *restricted branch multicast* (RBM). The main characteristics of this protocol are (1) a reduction in the fanout of a split packet, and (2) a scheme for deadlock *avoidance*, rather than deadlock detection and resolution.

In RBM, a packet may split into at most two paths at any site, one of which must be local—that is, it

includes the port connected to the computing node at that site. In our simulated implementation, the single non-local path is determined by the first non-local target in the packet, and *all* non-local targets are routed along that path. If there are any local targets, they may be split off. (If the port connected to the computing node is not available, the local targets must be routed down the non-local path.)

This restriction in itself does not prevent us from getting into deadlock—in figure 2, each split satisfies the conditions described above, yet deadlock occurs. To avoid deadlock, we provide an additional port, called the *split-port*, to the computing node at each site, to be used exclusively for the local path of a split packet. If this port is not available, the packet may not be split. It can either be buffered completely at the local site (using the normal unicast port to access the computing node), in which case the local targets will be stripped off and the remaining targets retransmitted later, or it must pass through without servicing any local targets.

From the point of view of the target site, the leaf of a multicast packet is identical to a unicast packet. Therefore, the split-port is not needed (and not used). Furthermore, the unicast port will eventually become free, since no multicast packet can use it and all packets are of finite size. Thus, no multicast leaves will block indefinitely, and deadlock is avoided. (If a packet is allowed to split using two or more non-local paths, as in RM, then two local ports are not sufficient to avoid deadlock.)

To calculate the average latency for RBM, we first assume that the distance between any pair of targets is $\bar{D}$. Second, we assume that the chances of encountering another target on the path is small—i.e., a packet is *only* split when it reaches the first target on its current target list. The head of the average RBM packet, then, will travel $\frac{1}{2}(n+1)\bar{D}$ hops to arrive at its target site. As in the RM case, the packet length must now take into account the additional targets. For the average packet, half the targets will have been split off when its destination is reached, giving a packet length of $\bar{L} + \frac{1}{2}nt$. Thus, the latency of the average packet

in an RBM multicast is given by

$$T_{\text{RBM}} = \frac{n+1}{2}\left\lceil\frac{t}{W}\right\rceil \check{D} + \left\lceil\frac{L}{W}\right\rceil + \frac{n-1}{2}\left\lceil\frac{t}{W}\right\rceil$$

$$= \left\lceil\frac{t}{W}\right\rceil \check{D} + \left\lceil\frac{L}{W}\right\rceil + \frac{n-1}{2}\left\lceil\frac{t}{W}\right\rceil + \frac{n-1}{2}\left\lceil\frac{t}{W}\right\rceil \check{D}.$$

Note that the only difference between this and the MU latency is the $O(nt\bar{D})$ term here, as opposed to the $O(n\bar{L})$ term in T_{MU}. Thus, if $\bar{L} > t\bar{D}$, RBM performs better than MU for a single multicast to randomly distributed targets. In the best case for RBM, all targets lie on the path between the source and the farthest target, in which case the total number of channels traversed equals the distance to the farthest target, and the average latency is the same as in the RM case.

2.4 Performance

According to the latency equations developed above, one would expect the RM scheme to be the clear winner. If we make the reasonable assumptions that $L \geq t$ and $\bar{D} \geq 1$, it will never have a higher latency than the other schemes. Those equations, however, assumed no contention in the network. To study how the three multicast schemes performed under load, we ran a test program called congest.

The congest program creates a number of processes, called *congestors*, each of which simultaneously multicasts a packet to a fixed number of other sites. While holding the network size and topology constant, there are three ways of increasing network load using congest:

- increasing the number of congestors,

- increasing the multicast fanout, and

- increasing the packet size.

The graphs in figure 3 show the effect of varying the number of congestors and packet size (data only) for two values of fanout (8 and 63). These experiments were simulated on an 8 x 8 torus. (Thus, a fanout of 63 corresponds to broadcast.) The congestors were placed randomly, one per site; the same random placement was used for all experiments. Finally, the multicast targets were picked randomly, and the order of the target list was also random.

Although the RM scheme predictably shows the lowest latencies for a single multicast, its performance degrades dramatically as the number of congestors is increased (figures 3a–b). This is not surprising, especially in the high fanout case, since simultaneous multicasts are very likely to interfere with one another, resulting in a high number of aborted packets.

The average latency for MU is almost constant for one, two, and four congestors—since there are few packets in the network at one time, there is little degradation due to network contention. The dominant cost in these cases is the time to place the multiple packets on the network. Finally, at sixteen congestors and above, the effects of network load become significant. Also, the average latency for MU is a strong function of packet size, especially for large fanout (figure 3c). This corresponds to the $O(n\bar{L})$ term in the MU latency equation.

The RBM scheme shows the lowest average latency in almost every multi-congestor configuration. It is competitive with RM for the single-multicast case, and clearly exhibits better performance under load.

With respect to MU, the effects of network contention are more pronounced for RBM under light loads. Each time an RBM packet must be temporarily stored and forwarded, the retransmission delay may effect more than one target (compared to only one target in MU). Thus, in congest, retransmission has a greater effect on the average latency than in MU. However, RBM still wins, except for very small packets, since it does not have the overhead cost of handling multiple packets at the source.

With small packets under high loads, MU shows lower average latencies than RBM. This is partly due to
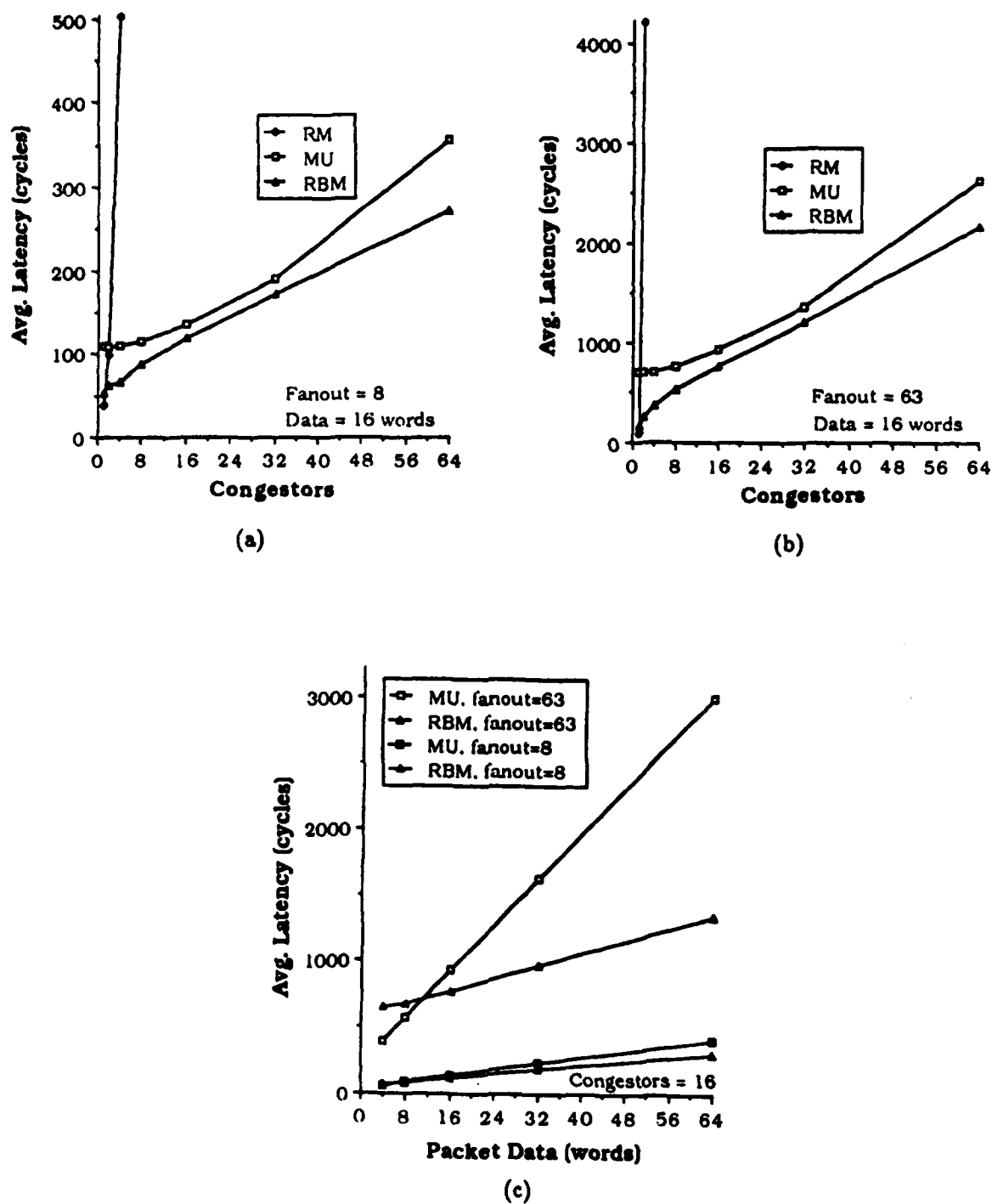
(a)

(b)

(c)

Figure 3: Average Latency running congest.

the impact of retransmission costs, discussed above. Additionally, the $O(n\bar{L})$ overhead of multiple packets is not as great when $\bar{L}$ is small. With smaller fanout (and thus lower load), the performance of MU and RBM are almost identical for small packets. Increasing the load effectively shifts the crossover point from $\bar{L} = t\bar{D}$ to some higher value of packet size.

A final distinction between MU and RBM is the rate at which the source can deliver messages to the network. In the MU scheme, the source must place n packets sequentially onto the network, so that no other packet may be sent for $O(n(\bar{L} + t))$ cycles. With RBM, the next message may be sent after $O(\bar{L} + nt)$ cycles. The rate at which messages may be sent to the network may limit the effective granularity of the computation at a site.

While it is interesting to study the behavior of these multicast protocols under varying loads, we recognize that congest is a contrived test case and may be pessimal, especially towards RM. In the next section, we will examine how the protocols perform in the context of a more realistic application program.

3   Performance with an Application

We studied the performance of the three multicast schemes in the context of ELINT[5], a real-time system for interpreting radar emissions from aircraft. The application correlates streams of radar emissions that have been observed by detection sites into radar emitters. These are grouped into clusters that are tracking together, and the activity of inferred aircraft is monitored. ELINT is implemented in a concurrent, object-oriented language called LAMINA[6], in which objects respond to messages by executing data-driven, run-to-completion tasks.

3.1 The Need for Multicast

ELINT uses pipelines of dynamically created objects to represent the hypothesized emitters and clusters in the airspace. Objects are replicated as necessary to widen pipelines and relieve congestion. This introduces the need to send the same data to multiple places, for example, in propagating timestamps to emitter pipelines, or in matching an emitter against a cluster, which is represented by replicated objects. ELINT relies on a multicast facility to provide this service.

3.2 Performance

For the experiment reported here, a typical object processed a 25–35 word message in 1500–2500 processor cycles before sending approximately one message of about the same size. The sources and sinks of messages were randomly distributed because objects were randomly sited as they were created. Multicasts constituted 7–9 percent of all transmissions and 25–30 percent of all receptions, with an average fanout of 4 and a maximum of 30 targets.

The simulations were performed on a 16 × 16 torus. The network latency distributions of multicasts and unicasts, measured upon arrival at the targeted sites, is shown in figure 4.

Multicasts performed best with the restricted branch scheme, achieving an average latency that was one-third that of multi-unicasts. Resumable multicasts performed very poorly, showing an average latency that was more than an order of magnitude higher than the others. (This average may be unduly low, as discussed later.) Although 40 percent of both restricted branch multicasts and multi-unicasts were delivered within 120 communication cycles, 95 percent of the former were delivered within 400 cycles versus 75 percent of the latter.

The majority of unicasts displayed virtually identical performance with both multi-unicast and restricted branch multicast, although the average latency was half as much with the latter because of the smaller tail
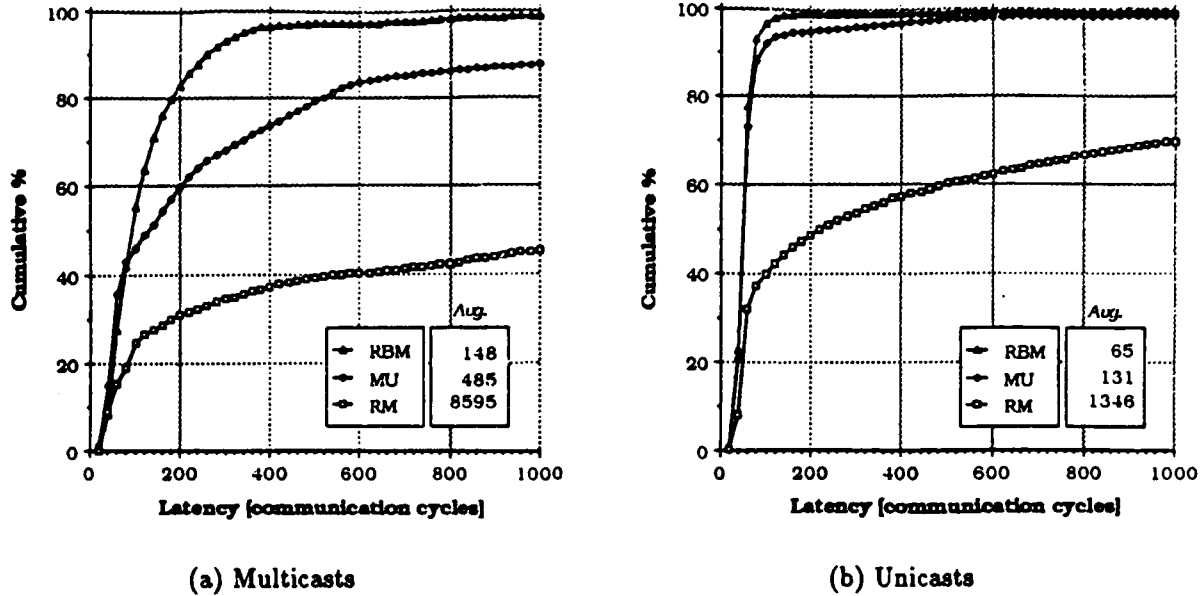
(a) Multicasts

(b) Unicasts

Figure 4: Latency distributions with ELINT; 16 × 16 torus.

in the distribution. Resumable multicasts, on the other hand, significantly impacted unicasts and increased their average latency by an order of magnitude.

To understand why this is so, let us consider each multicast approach in turn.

- **Multi-Unicast:** The $O(n\bar{L})$ source delay involved in injecting multicasts into the network is the major drawback of MU. Besides affecting the constituent messages of the multicast as discussed in section 2.4, it also impacts messages that are queued behind it, be they messages initiated by the computing node at that site or those that are temporarily stored at the site due to network congestion. In ELINT, $\bar{L}$ was sufficiently large and multicasts sufficiently frequent for this to be visible in the latency of both multicasts and unicasts.

- **Resumable Multicast:** Multicasts transmitted with RM both increase network load as well as perform badly in the presence of such load. As we found in section 2.4, their branching policy increases

the probability of their being aborted and retransmitted; additionally, it makes them consume excessive network resources. Thus, they impede their own progress as well as that of other messages in the network.

This was particularly evident in ELINT, for two reasons. First, some high-fanout multicasts were driven by the arrival of input data (*e.g.*, the distribution of timestamps to emitters); the constant data arrival rate led to the generation of multicasts at a constant rate that was higher than the rate at which they could be delivered. Second, multicasts often occurred in bursts. The net result was the same in both cases—increasingly severe contention for limited resources, to the detriment of all. In fact, the only reason many multicasts were successfully delivered with RM was that the program accepted only a finite amount of data so that new multicasts stopped being generated, allowing those in transit to finally complete. Hence, the "average" latency of RM multicasts reported here may be unduly optimistic.

• **Restricted Branch Multicast:** There are two potential drawbacks with the RBM approach. The first is the $O(n\bar{D})$ latency that is incurred as the front edge of a multicast visits each target in turn. For the 16 × 16 torus, $\bar{D}$ is approximately 8; this was sufficiently small relative to $\bar{L}$ to make this latency tolerable.

The second problem is that the delay incurred by a multicast packet that is stored and forwarded due to network congestion is also incurred by all its constituent messages (targets). In ELINT, 5–7 percent of all packets (multicasts and unicasts) were stored and forwarded for both RBM and MU. This affected twice as many messages using the RBM scheme. However, the average latency that the MU packets experienced due to this was by far the higher, because the site resources which handled storage and forwarding were also more heavily utilized in transmitting multi-unicasts. Thus, in fact, network load had less overall impact with RBM than with MU.

4 Conclusions and Future Work

The experiments reported here represent only the first step toward a complete evaluation of the costs and benefits of network-supported multicast. More work is needed to understand the performance of these protocols for applications and systems with different network load characteristics, such as the multicast invalidate traffic of a directory-based cache coherence scheme. Additionally, a detailed analysis of the hardware cost is required. Finally, performance enhancements should be considered, such as sorting the targets of an RBM packet to specify an optimal path or adding resources to reduce the cost of retransmitting temporarily buffered packets.

Nevertheless, the data presented here suggests that multicast transmission can be directly supported in multicomputer systems which utilize communication networks based on cut-through routing with minimal buffering. The restricted branch multicast protocol provides lower latency than a multi-unicast approach, over a significant range of network conditions. While the resumable multicast protocol provides good performance in isolation, its behavior in the presence of network contention appears to make it unsuitable for general use.

Acknowledgements

References

[1] Gregory Byrd, Russell Nakano, and Bruce Delagi. A dynamic cut-through communication protocol with multicast. Technical Report KSL-87-44, Knowledge Systems Laboratory, Stanford University, August 1987.

[2] Gregory T. Byrd and Bruce A. Delagi. A performance comparison of shared variables *vs.* message passing. In *3rd Annual International Supercomputing Conference*. Information Sciences Institute, May 1988.

[3] William J. Dally and Charles L. Seitz. The torus routing chip. *Distributed Computing*, 1:187–196, 1986.

[4] William J. Dally and Charles L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.

[5] Bruce A. Delagi and Nakul P. Saraiya. ELINT in LAMINA: Application of a concurrent object language. Technical Report KSL-88-33, Knowledge Systems Laboratory, Computer Science Department, Stanford University, October 1988.

[6] Bruce A. Delagi, Nakul P. Saraiya, and Gregory T. Byrd. LAMINA: CARE applications interface. Technical Report KSL-86-67, Knowledge Systems Laboratory, Computer Science Department, Stanford University, November 1987.

[7] I. S. Gopal. Prevention of store-and-forward deadlock in computer networks. RC 10677, Research Division, International Business Machines Corporation, 1984.

[8] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3:267, 1979.

END

FILMED

7-89

DTIC